CA-1: CASE STUDY

AUTOMATED STOCK TRADING SYSTEM

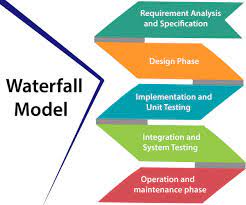
Performed by :

1. Ayush Kumar 22070122041
2. Aryan Tame 22070122034

Github repository link - https://github.com/Aryant04/Flexi

We will be progressing through the life cycle of the project using waterfall model

Waterfall model:



As shown above, the first step towards the completion of the project is requirement gathering. We will be creating a SRS (Software Requirement Specification) Document for a thorough understanding of the requirements which need to be fulfilled in order to complete the project.

SRS DOCUMENT

**Software Requirement Specification (SRS) Document** **for the Automated Stock Trading System**

**Version:** 1.0

**Date:** 31st August 2023

**Prepared by:** Ayush Kumar And Aryan Tambe

**Client:** Shri Sasarwati Ltd.

**Project:** Automated Stock Trading System (ASTS)

**Table of Contents**

1. **Introduction**

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms, and Abbreviations

1.4 References

1.5 Overview of the Document

1. **Overall Description**

2.1 Product Perspective

2.2 Product Features

2.3 User Classeses

1. **Specific Requirements**

3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.2 Hardware Interfaces

3.1.3 Software Interfaces

3.2 Functional Requirements

3.2.1 User Registration and Authentication

3.2.2 Stock Data Retrieval

3.2.3 Algorithmic Trading Engine

3.2.4 User Portfolio Management

3.2.6 Reporting and Notifications

3.3 Non-Functional Requirements

3.3.1 Performance

3.3.2 Security

3.3.3 Usability

* + 1. Scalability

**4 . Use case diagrams:**

**1.INTRODUCTION**

**1.1 PURPOSE:**

The purpose of this document is to provide a comprehensive overview of the requirements for the development of the Automated Stock Trading System (ASTS). It outlines the functional and non-functional requirements, system constraints, and assumptions.

**1.2 SCOPE:**

The ASTS is designed to automate stock trading activities by utilizing advanced algorithms and real-time market data. It is intended for use by experienced traders and investors to execute trades efficiently and mitigate risks

**1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS:**

* ASTS: Automated Stock Trading System
* API: Application Programming Interface
* GUI: Graphical User Interface

**1.4 REFERENCES :**

<https://github.com/RaviTambade>

**1.5 OVERVIEW OF THE DOCUMENT**

This document is structured to provide a clear understanding of the requirements for the ASTS. Section 2 presents the overall description of the system. Section 3 details the specific requirements, including external interfaces, functional and non-functional requirements, and system use cases. Section 4 discusses the planned system evolution and enhancements.

**2. Overall Description**

**2.1 Product Perspective:**

The ASTS is a standalone software system that interacts with external stock market data providers and trading platforms through APIs.

**2.2 Product Features:**

**The ASTS is an automated trading system, so it can effectively trade based on user defined strategies, without human intervention. User can also create their own custom strategies and can backtest and verify their results before staking any of their own money.**

**2.3 User Classes:**

**1. Traders: They are experienced in stock trading and require advanced features for executing trades, analysing market data and managing portfolios.**

**2. Administrators:** Administrators have elevated privileges and responsibilities within the system. They are responsible for system management, user access control, and configuration. In the case of a trading system, administrators might oversee user accounts, set trading rules, and ensure system security.

**3. Developers:** In some cases, software developers or IT personnel may use the system for maintenance, troubleshooting, or enhancements. They might require special access for debugging or making system updates.

**2.5 Design and Implementation Constraints:**

1. **Hardware Constraints:**

* These constraints involve limitations related to the hardware infrastructure on which the software will run.
  + Minimum and maximum hardware specifications (e.g., processor speed, RAM, storage).
  + Compatibility requirements with specific hardware components or devices.

2. **Software Constraints:**

* Software constraints refer to limitations imposed by the software environment in which the system will operate.
  + Compatibility with specific operating systems (e.g., Windows, Linux, macOS).
  + Dependency on third-party software components or libraries.
  + Restrictions on software versions or platforms that the system must support.

3. **Legal and Regulatory Constraints:**

* Legal and regulatory constraints encompass compliance with laws, regulations, and industry standards.
  + Data privacy laws (e.g., GDPR, HIPAA) that dictate how user data must be handled.
  + Financial regulations governing trading activities.

**3. Specific Requirements**

**3.1 External Interface Requirements:**

**3.1.1 User Interfaces: The** SRS should provide a clear and concise definition of what the user interface encompasses within the software. This includes graphical user interfaces (GUIs), command-line interfaces (CLIs), web-based interfaces, mobile app interfaces, or any other method through which users interact with the system.

Guidelines include:

* Consistency in design elements such as buttons, fonts, and color schemes.
* Responsiveness and compatibility with different screen sizes and devices (for web and mobile interfaces).
* Usability standards, including intuitive navigation and user-friendly layouts.
* Accessibility considerations to ensure that the UI is usable by individuals with disabilities.

**User Roles and Permissions:**

* **End-user: The users who will use the application to trade.**
* **Admin:** They are responsible for system management, user access control, and configuration. In the case of a trading system, administrators might oversee user accounts, set trading rules, and ensure system security.
* **Developers:** In some cases, software developers or IT personnel may use the system for maintenance, troubleshooting, or enhancements. They might require special access for debugging or making system updates.
* **BOD(Board Of Directors):** Special permissions to view user accounts, and specialized portfolio to view and analyse performance of the product.

**Navigation and Menus:**

Users can easily move between different sections or features of the software.

**User Feedback and Notifications:**

This includes notifications, alerts, and messages that inform users of system status, errors, or important updates.

**3.1.2 Hardware Interfaces :**

**Types of Hardware Interfaces:**

* + Input/Output (I/O) Interfaces: Connections to devices for data input (e.g., sensors) and output (e.g., displays).
  + Communication Interfaces: Interfaces for data exchange with external systems or devices (e.g., USB ports, serial ports, network interfaces).
  + Storage Interfaces: Connections to storage devices such as hard drives, SSDs, or external storage media.
  + Peripheral Interfaces: Interfaces for connecting peripherals like printers, scanners, and cameras.

**Data Transfer Protocols:**

* USB, Bluetooth, Wi-Fi, Ethernet, RS-232, and custom communication protocols.

**3.1.3 Software Interfaces :**

These include the SDK(software development toolkit), the correct compilers for the languages used(in this case Python), the APIs used, we used kiteconnect API which is the developer’s API for Zerodha.

**3.2 Functional Requirements:**

**3.2.1 User Registration and Authentication:**

**The user must be able to log into the broker’s website directly through the ASTS to be able to access the website’s services and to be able to place trades automatically. By entering the username and password inside the system once, the user should be able to automatically log into the system and proceed without having to sign in every time he wants to use the interface. So, the credentials should be saved for further use.**

**3.2.2 Stock Data Retrieval:**

**The user should be able to retrieve historical data through the API of the broker so that he can devise strategies based on the data and back-test and validate his strategies using the data retrieved.**

**3.2.3 Algorithmic Trading Engine:**

**The system should have an algorithmic trading engine that can decide when to enter and when to exit a trade using the strategies defined by the user. It should have an extensive database of strategies and should be capable of allowing the user to define his own custom strategies.**

**3.2.4 Portfolio Management:**

**The system should be able to display and correctly update the user’s portfolio every time a trade is executed.**

**3.2.5 Reporting and Notifications:**

**The system should report and notify, preferably with a loud sound, so that the user is alerted and can monitor the trades that are being executed.**

**3.3 Non-Functional Requirements**

**3.3.1 Performance:**

**The system should perform efficiently and be fast during its operations. It should be able to review the back-test result and compare it with the current market to modify and recommend better parameters for the trades to be executed.**

**3.3.2 Security:**

**The system has access to the user’s funds for trading; hence it is required for it to be safe to ensure the safety of the user’s money. The data has to be encrypted and the access protocols should be strict so that nobody else apart from the user can log into their accounts.**

**3.3.3 Usability:**

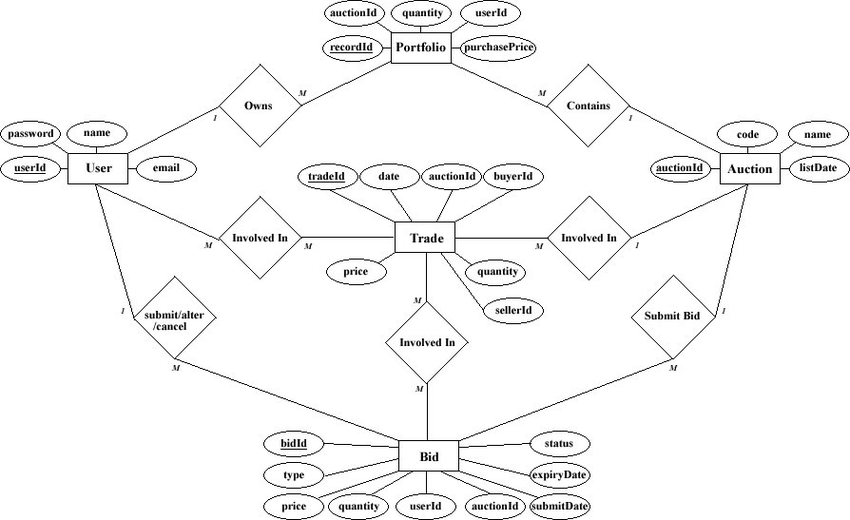
**The system should be user friendly, so that people who are new do not lose their money before understanding the risks.**

**3.3.4 Scalability:**

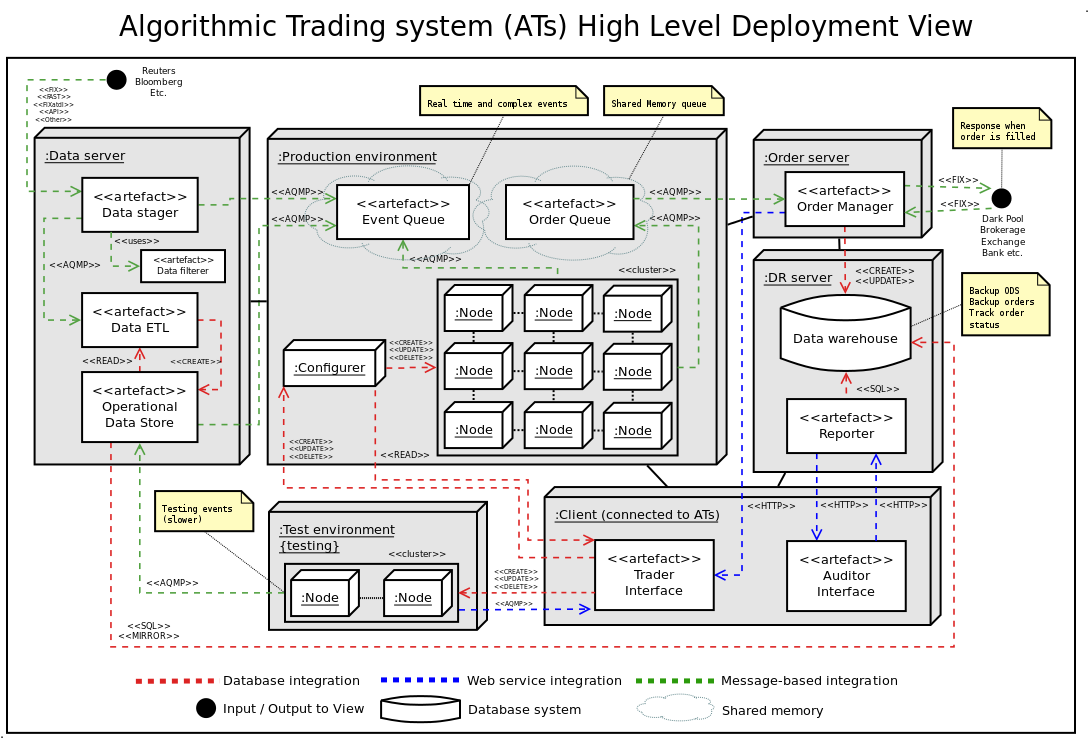
**The system should be scalable and should adapt according to the user’s needs and with the broker’s website as well.**

1. **USE CASE DIAGRAMS :**

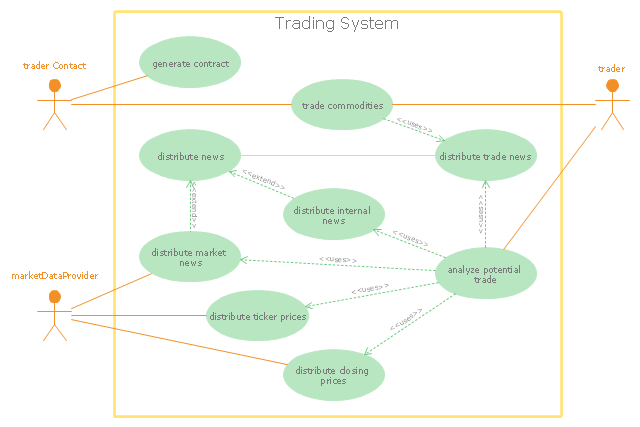
**Entity Relationship diagram -**



**Deployment diagram:**



**USE CASE DIAGRAM-**



**CLASS DIAGRAM –**

A computer screen shot of a diagram

Description automatically generated